EXPERIÊNCIA 5:

IMPLEMENTAÇÃO DE RELÓGIO DIGITAL COM INTERRUPÇÃO

Autores: Prof. Dr. Marco Túlio Carvalho de Andrade, Prof. Dr. Carlos Eduardo Cugnasca, Prof. Dr. André Riyuiti Hirakawa

Colaborador: Prof. Dr. Jorge Kinoshita

Versão 1.5 - 05/2007

1. OBJETIVO

Esta experiência tem como objetivo a familiarização com os mecanismos de interrupção suportados pelo microcontroladores 80C51, utilizado na Placa Experimental.

Pretende-se que ao final desta o aluno seja capaz de habilitar interrupções, projetar rotinas de tratamento de interrupções associando-as aos endereços pertinentes, configurar periféricos para geração de interrupções e utilizar uma interface de entrada e saída programável para a apresentação dos resultados: o módulo do display de cristal líquido.

2. MECANISMOS DE TRANSFERÊNCIA DE DADOS

Os sistemas baseados em microprocessadores e microcontroladores requerem a interligação com diversos dispositivos periféricos, tais como portas de entrada/saída paralelas, canais de comunicação seriais, controladores de teclado e display, conversores A/D, temporizadores, contadores de eventos, etc. Alguns dispositivos de entrada e saída, como o módulo de display de cristal líquido, que já possuem internamente os principais circuitos de controle, sendo simples o seu interfaceamento com um processador. Contudo, outros periféricos requerem tratamento especializado, envolvendo ações em instantes de tempo periódicos, enquanto que outros são utilizados em aplicações que exigem pronto atendimento do microprocessador em instantes aleatórios, como por exemplo, sistemas e equipamentos projetados para operar em **tempo real**. No caso da ocorrência de eventos simultâneos, os mecanismos normalmente disponíveis em um permitem que se priorize o mais importante.

Atualmente, para se implementar as principais funções de entrada e saída encontram-se disponíveis circuitos integrados programáveis. Muitos desses componentes desempenham boa parte das funções requeridas para a transferência de dados, simplificando as atividades do processador, e conferido, assim, melhor desempenho ao conjunto. Algumas arquiteturas utilizam processadores específicos de entrada e saída, visando a distribuição das atividades para se obter um melhor desempenho do conjunto. Por exemplo, um microcontrolador dedicado para controlar o teclado de

um microcomputador pessoal libera o processador principal, de forma que ele possa ter um melhor aproveitamento nas atividades de execução de programas.

Para se implementar a transferência de dados entre os microprocessadores e os dispositivos periféricos são utilizadas diversas técnicas, escolhidas em função das particularidades de cada aplicação. Algumas das principais técnicas são relacionadas a seguir:

a) Transferência Incondicional

Consiste na execução da operação de entrada ou saída no instante em que o microprocessador puder ou desejar, sem a verificação da disponibilidade do dado ou da viabilidade de se executar tal tarefa. É utilizada quando o dispositivo periférico não requer tratamento especial.

Exemplos: coleta de informações do estado de chaves externas modificadas com pouca freqüência, como as de configuração, envio de programação ou comandos aos dispositivos periféricos, leitura de palavras de estado de dispositivos periféricos, envio de informações de sinalizações para leds, lâmpadas e displays em interfaces sem multiplexação, etc.

b) Transferência Condicional

Também conhecida por ”**Wait-for-Flag**”, consiste na execução da operação de entrada ou saída condicionada à ocorrência de outro evento externo, ou à disponibilidade do dado (*Figura 1.a).*

O microprocessador normalmente executa um *loop* de programa, efetuando a leitura da porta de entrada que fornece a informação da ocorrência ou não de tal evento, ou informando que o dado está disponível para leitura (*status* do periférico, ou ***flag***). Caso ele tenha ocorrido, a transferência é efetuada, encerrando-se o *loop*. Esse método apresenta como inconvenientes:

- o bloqueio do processamento durante a execução do *loop*;
- o consumo de tempo útil de trabalho;
- a maior dificuldade de tratamento de mais de um evento, principalmente no que diz respeito à atribuição de prioridades a eles, e à elaboração do software.

Em algumas aplicações que não requerem pronto tratamento, é possível intercalar algum processamento entre leituras de palavras de *status*. As principais aplicações desse método correspondem a situações em que o microprocessador nada tem a fazer enquanto o dispositivo periférico não sinalizar a ocorrência do evento (que é o caso de muitos equipamentos com interface com o operador), comunicações paralelas com sincronismo e altas velocidades, etc. Exemplo: o módulo do display de cristal líquido utilizado na Placa Experimental apresenta normalmente velocidade menor de processamento de informações do que microcontrolador; o uso de seu Busy Flag possibilita a compatibilização das velocidades, segundo a técnica descrita.

c) Transferência Programada/Amostragem

É uma variação da transferência condicional, quando o evento externo tem um intervalo de tempo fixo e grande quando comparado com a velocidade do microprocessador. Nesse caso, as transferências são efetuadas periodicamente, com o microprocessador controlando o intervalo de tempo (*Figura 1.b*).

Como inconveniente tem-se a necessidade de ajustar o intervalo de tempo por software, muitas vezes em função do tempo de execução do programa, e que nem sempre garante precisão desejada para muitas aplicações de controle. Para se resolver esse problema, procura-se garantir uma maior precisão do ***intervalo de amostragem*** t_a, através do uso de interrupções periódicas, determinadas por um oscilador externo.

Trata-se de uma técnica muito utilizada em equipamentos destinados a supervisão e controle de processos, uma vez que os algoritmos de controle obrigam a retirada de amostras digitalizadas dos sinais e atuação em intervalos de tempo iguais (Δt).

Exemplos: aquisição de dados de conversores A/D, saídas pulsadas, varredura de teclados, multiplexação de displays, etc.

d) Interrupção

Para se tornar mais eficiente o tratamento de periféricos que solicitam operações do microprocessador assincronamente, ou até mesmo periodicamente, existe o recurso denominado **interrupção**. Ele requer sinais que, em geral, todo microprocessador apresenta, e eventualmente algum circuito externo adicional. Utilizando os sinais de entrada de interrupção do microprocessador, os dispositivos periféricos podem informar a ocorrência de um dado evento externo, que será tratado com maior facilidade e rapidez através das **subrotinas de tratamento de interrupção**, em geral sem muito comprometer o programa que estava em execução: as condições internas do microprocessador são preservadas para posterior continuação da execução do programa interrompido (*Figura 1.c*). Com esse método, o microprocessador não necessita consumir tempo de processamento para pesquisar a ocorrência de eventos externos e pode garantir um tempo reduzido para iniciar o seu tratamento.

Exemplos: sistemas com muitos dispositivos periféricos, uso de dispositivos cujos eventos envolvidos são sempre sinalizados assincronamente (canais de comunicação serial ou paralelos, temporizadores e contadores programáveis, controladores de periféricos, relógios de tempo real, sinais de emergência, sinais de sensores digitais - pulsos, nível, etc).
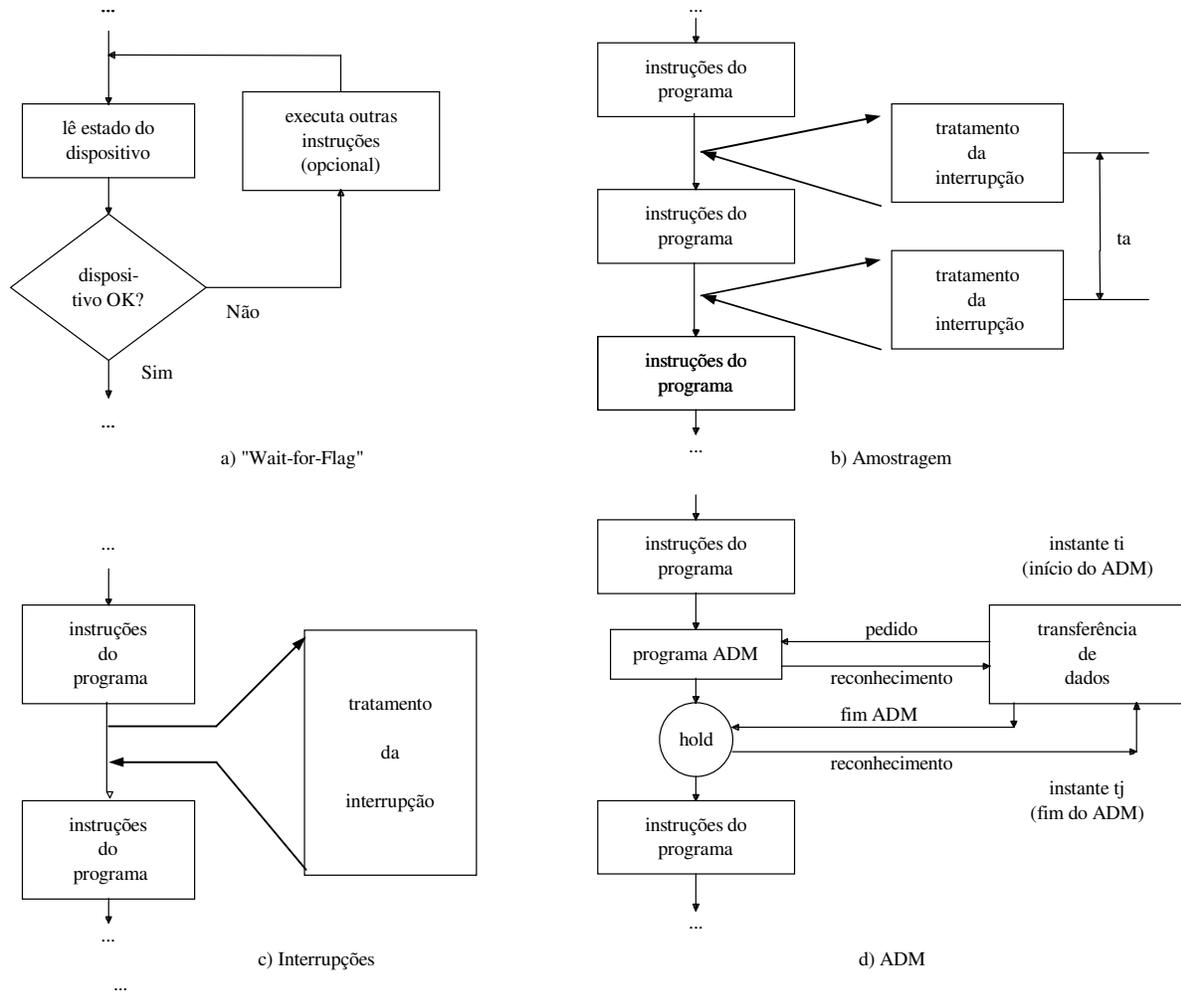
Figura 1 - Principais Técnicas de Transferência de Dados de Entrada e Saída.

e) Transferência Via Acesso Direto à Memória (ADM)

Utilizada normalmente quando as informações a serem transferidas se encontram dispostas seqüencialmente na memória, e a sua quantidade é muito grande. Esse método requer um circuito adicional denominado Controlador de ADM que é programado pela UCP, e que efetua automaticamente a transferência de dados entre a memória e o dispositivo periférico, ou até mesmo outra memória, sem a presença do microprocessador (que fica bloqueado durante a operação de ADM, no estado conhecido como ***hold***, não podendo realizar qualquer atividade uma vez que as vias estão sendo utilizadas para a transferência dos dados), tornando-a mais rápida (*Figura 1.d*).

Exemplos de dispositivos periféricos que comumente se utilizam dessa técnica: controladores de disco, canais de comunicação de alta velocidade, etc.

3. MECANISMOS DE INTERRUPÇÃO

3.1. Características Básicas de Interrupções

Como foi apresentado, as interrupções possibilitam um tratamento mais eficiente das operações de entrada e saída, permitindo melhor atender aos requisitos de aplicações em tempo real.

A requisição de interrupção pode ocorrer a qualquer momento (assincronamente), sendo ela indicada pela ativação de um *flag* pelo dispositivo periférico. O microprocessador reconhece a interrupção, enviando sinais de controle, completa a execução da instrução corrente, salva o conteúdo dos registradores de interesse (contador de programa, registrador de estado, etc), e atende à solicitação do dispositivo periférico que solicitou a interrupção, transferindo o controle para a subrotina de tratamento da interrupção. Ao término da execução desta, o *flag* de indicação de interrupção é desligado, os registradores que foram salvos são restaurados, e o controle do programa de dirigido para a instrução seguinte ao ponto de interrupção do programa.

Algumas interrupções podem ter seu tratamento postergado, enquanto que outras necessitam de tratamento imediato (por exemplo, sinais de emergência). Em função disso, a maioria dos microprocessadores apresentam interrupções **mascaráveis** e interrupções **não-mascaráveis**. Através de instruções apropriadas o programa pode habilitar ou desabilitar uma interrupção mascarável, enquanto que a não-mascarável deverá ser sempre atendida, devendo ser reservada, assim, apenas para eventos de alta importância (por exemplo, sinal de emergência de uma máquina ou equipamento). Assim, uma aplicação de tempo real que envolve módulos de programas críticos e que não podem ser interrompidos durante sua execução, deve utilizar interrupções mascaráveis.

Outra característica das interrupções diz respeito à forma de obtenção dos endereços das subrotinas de tratamento, que varia de um microprocessador para outro. Ela costuma ser do tipo ***fixa***, ***vetorada*** ou ***não-vetorada***:

- **Interrupções fixas:**

  Requerem um hardware relativamente simples de interface. Um *flag* é ativado pela linha de interrupção, indicando a requisição. Caso apenas um dispositivo periférico esteja ligado a essa linha, então o controle é transferido para uma posição fixa de memória, que apresenta a subrotina de tratamento da interrupção. Quando vários dispositivos são acoplados a uma mesma linha, cada um com sua própria subrotina de tratamento, o microprocessador tem que identificar o dispositivo periférico responsável pela interrupção. Para contornar essa dificuldade, podem-se utilizar múltiplas linhas, uma para cada interrupção, tendo-se uma posição fixa de memória para a colocação da subrotina de tratamento de cada uma. Entretanto, existe a necessidade de tantas linhas quantos forem os dispositivos periféricos que solicitam interrupção. Exemplos: interrupções RST5.5, RST6.5, RST7.5 E TRAP do 8085, interrupção NMI/ do Z80.

- **Interrupções vetoradas:**

  Este tipo exige a identificação do dispositivo periférico que solicita interrupção. Essa identificação é utilizada para a localização do endereço da subrotina de tratamento da interrupção, em uma tabela localizada em uma região determinada da memória do **microprocessador (vetor de interrupções**). Exemplo: interrupções da família MC68000.

- **Interrupções não-vetoradas:**

  Nesse tipo, o dispositivo periférico fornece ao microprocessador diretamente o endereço da subrotina de tratamento da interrupção (por exemplo, as interrupções do 8086/8088). Em alguns sistemas, o dispositivo periférico fornece apenas metade de endereço, estando a outra metade armazenada em um registrador do microprocessador, carregado na fase de inicialização do sistema (por exemplo, as interrupções do Z80 operando no modo 2). Em outros sistemas, o dispositivo periférico tem que fornecer uma instrução ao microprocessador, normalmente a instrução de chamada da subrotina de tratamento da interrupção (por exemplo, a interrupção INTR do 8085, e INT/, do Z80 operando no modo 0).

  Em sistemas com mais de uma interrupção, existe a possibilidade de ocorrerem pedidos de interrupção simultâneos, devendo existir algum critério para a escolha de qual será atendida em primeiro lugar. Esse critério, normalmente por prioridade, pode ser implementado de diversas formas: *daisy chain* ou circuitos de prioridade.

  - ***Daisy chain*:**

    Nesse esquema, os dispositivos periféricos que podem solicitar interrupção são interligados (em cascata), através de sinais de controle, em uma cadeia conhecida como *daisy chain*, sendo o primeiro elemento ligado ao microprocessador. Em caso de pedido de interrupção, o microprocessador envia um sinal ao primeiro elemento; caso seja ele o autor do pedido, ele responde com o endereço da subrotina de tratamento; caso contrário, ele repassa o pedido ao próximo dispositivo periférico, que repetirá o procedimento descrito. Dessa maneira, o primeiro elemento da cadeia deverá corresponder ao de maior prioridade, e assim sucessivamente. Esse esquema pode ser implementado, por exemplo, com o Z80 operando no modo 2.

  - **Circuitos de prioridade:**

    Nesse esquema, um codificador de prioridades é utilizado (por exemplo, de 8 para 3), fornecendo em sua saída o código do pedido da interrupção de maior prioridade presente na entrada. Esse código pode ser lido pelo microprocessador para descobrir qual interrupção atender. Muitas vezes são disponíveis esquemas para o mascaramento individual das interrupções, permitindo que uma interrupção alocada em uma linha de maior prioridade não

seja atendida quando outra de menor prioridade ocorrer. Esse esquema pode ser implementado, por exemplo, com 8085, Z80 operando no modo 0, e a família MC68000.

Muitos processadores apresentam uma de suas interrupções como sendo **não-mascarável**, de forma que sempre que ela ocorrer, o microprocessador deverá atendê-la. Ela deve ser utilizada com muito cuidado, devendo ser alocada a eventos de muita importância, como por exemplo, queda da alimentação, sinal de emergência, alarme, etc.

3.2. Interrupções na Família 80C51

Cada família de microprocessadores apresenta um tipo de implementação para as suas interrupções, sendo encontradas muitas particularidades não necessariamente encontradas em outras famílias. Recomenda-se, nesse ponto, que seja efetuada uma pesquisa sobre as interrupções do 80C51 em alguma(as) das referências: [17], [16] (Application Builder), [13], [14], [15], Capítulo 2 de [1], Nota AN420 de [2], Capítulo 6 de [8], Capítulo 6 de [9], Páginas 10 e 11 de [7] no tocante aos nomes de bits e registradores para configurações em geral. A seguir, serão resumidos os principais conceitos envolvidos.

As interrupções do 80C51 são vetoradas, com vetor fixo na memória de programa. Cinco fontes de interrupção são disponíveis: duas interrupções externas (**EX0** – pino **INT0/** – **P3.2** e **EX1** – pino **INT1/** – **P3.3**), duas interrupções por *timer* (**ET0** e **ET1** – internas) e uma interrupção do canal serial (ES – interna). Em ordem de prioridade tem-se, da mais prioritária para a menos prioritária: **EX0**, **ET0**, **EX1**, **ET1** e **ES**. A interrupção de prioridade maior interrompe a interrupção de prioridade menor. O **mascaramento**, ou desabilitação específica de uma interrupção, é individual, através dos bits do registrador **Interrupt Enable (IE) - A8H**, representado a seguir, sendo que a habilitação e desabilitação global é feita através do seu bit 7 – **EA** (valor 0 desabilita, valor 1 habilita); após o reset, o seu valor é 0.

BIT	IE.7	IE.6	IE.5	IE.4	IE.3	IE.2	IE.1	IE.0
SINAL	EA			ES	ET1	EX1	ET0	EX0
END.	AFH	AEH	ADH	ACH	ABH	AAH	A9H	A8H

Cada interrupção pode ter um ou dois níveis de prioridade, de acordo com uma inicialização colocada em um bit do registrador **Interrupt Priority (IP) – B8H** [1], representado a seguir (valor 0 - interrupção de baixa prioridade, valor 1 - interrupção de alta prioridade). Uma interrupção de prioridade maior não pode ser interrompida, mas pode interromper uma interrupção de prioridade menor. Uma interrupção de baixa prioridade somente pode ser atendida se nenhuma outra estiver sendo tratada. Dois grupos de prioridade (*nesting*) podem ser definidos.

BIT	IP.7	IP.6	IP.5	IP.4	IP.3	IP.2	IP.1	IP.0
SINAL				PS	PT1	PX1	PT0	PX0
END.	BFH	BEH	BDH	BCH	BBH	BAH	B9H	B8H

As interrupções externas **INT0** e **INT1** podem ser programadas como sensíveis a nível 0 ou sensíveis à borda de descida, dependendo da programação dos bits **IT0** e **IT1** no registrador **TCON (88H)**, representado a seguir. Os flags gerados com estas interrupções estão disponíveis nos bits **IE0** e **IE1** desse registrador.

BIT	TCON.7	TCON.6	TCON.5	TCON.4	TCON.3	TCON.2	TCON.1	TCON.0
SINAL	*TIMERS 0 e 1*				IE1	IT1	IE0	IT0
END.	8FH	8EH	8DH	8CH	8BH	8AH	89H	88H

- **ITi = 1**: a **INTi/** é reconhecida na transição de 1 para 0 do sinal, que deve permanecer em 0 por pelo menos 12 períodos de *clock*.
- **ITi = 0**: a **INTi/** é reconhecida se na amostragem das interrupções pelo microcontrolador ela estiver em 0 (deve voltar para 1 antes do retorno da rotina de tratamento.
- **IEi**: o hardware interno leva o sinal para 1 quando detectada uma transição de 1 para 0 em **INTi/**, e leva o sinal para 0 quando trata **INTi/.**

A Figura 2, resume a estrutura de interrupções do 80C51 e 80C52 (este possui duas interrupções adicionais).
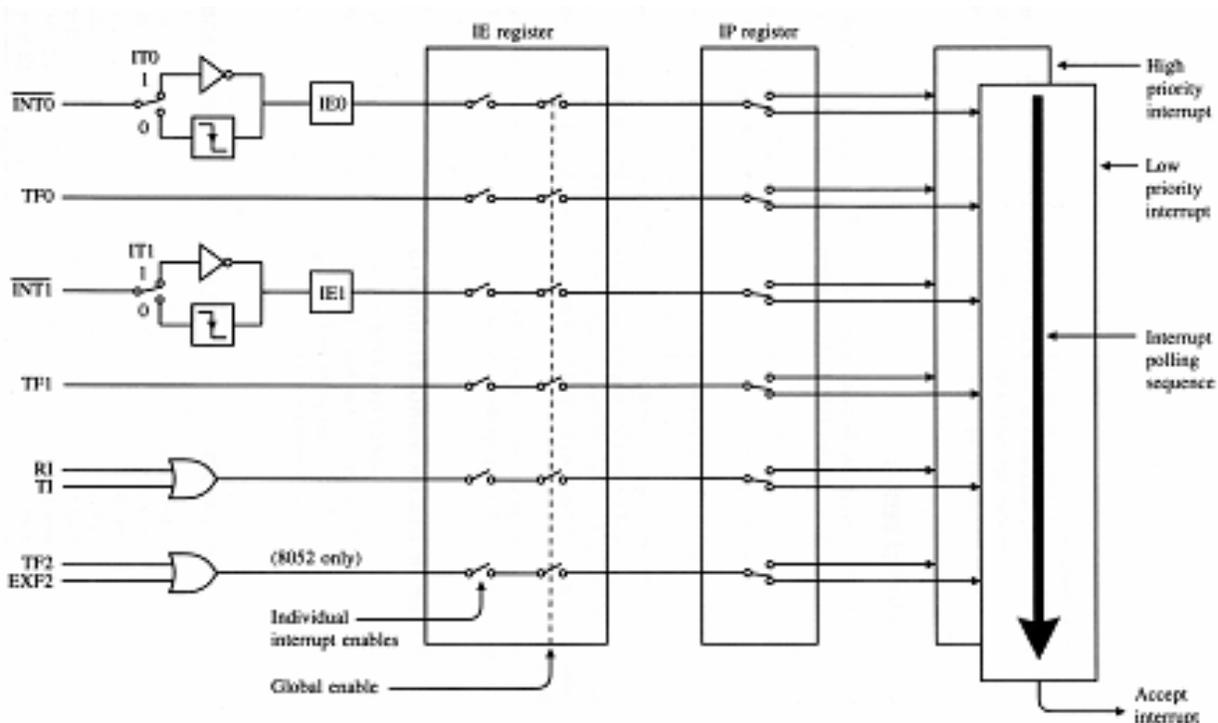
Figura 2 – Estrutura das interrupções do 80C51/80C52. Extraída de [15].

O tratamento de interrupções no 80C51 prevê endereços específicos e consecutivos, para onde o processamento é transferido para cada ocorrência de interrupção. Esses endereços se localizam na região inicial da área de programa. Como essa área encontra-se em memória ROM, instruções de desvio (**LJMP** - pulo incondicional) para a área de RAM, foi implementado visando permitir que o usuário possa, a partir destes endereços, implementar e associar os programas de tratamento para cada interrupção através do Programa Monitor (por exemplo, na posição **000BH**, área que possui uma memória apenas de leitura – EPROM, foi colocada a instrução **LJMP FFF3H**: nessa posição o usuário deve colocar uma outra instrução de desvio incondicional para o endereço da rotina de tratamento da interrupção do Timer 0). Os endereços de desvio de cada interrupção são apresentados na Tabela I.

Assim, os programas de tratamento de interrupções devem ser colocados em local de memória conhecido, e os desvios para eles devem ser colocados nos endereços correspondentes, de acordo com a interrupção gerada.

Tabela I – Endereços para Tratamento de Interrupção

Interrupção	Endereço do Vetor de Interrupções (ROM)	Endereço de uma Cópia do Vetor de Interrupções na RAM	Origem da Interrupção
EX0	**0003H**	**FFF0H**	Interrupção externa 0
ET0	**000BH**	**FFF3H**	Timer/Contador 0
EX1	**0013H**	**FFF6H**	Interrupção externa 1
ET1	**001BH**	**FFF9H**	Timer/Contador 1
ES	**0023H**	**FFFCH**	Canal Serial

Quando uma interrupção ocorre e é aceita, o microcontrolador: automaticamente coloca na Pilha o **Program Counter (PC)** – primeiro o byte menos significativo; bloqueia as interrupções de igual ou menor prioridade; no caso de interrupções dos Timers ou externas, os respectivos *flags* são limpos; a execução do programa é transferido para o endereço da respectiva posição da interrupção no **Vetor de Interrupções**, onde se espera que tenha sido colocado um pulo incondicional para o início da subrotina de tratamento de interrupção.

Para o tratamento de cada interrupção devem ser desenvolvidas subrotinas específicas, cuja última instrução deverá ser **RETI –** retorno de interrupções (diferente de **RET** – retorno de uma subrotina comum). Essa instrução provoca a retirada da pilha dos dois bytes do endereço de retorno para o programa principal interrompido, e a restauração do estado das interrupções.

Qualquer rotina de tratamento de interrupção apresenta a seguinte estrutura:

nome: **Salva na pilha os registradores que a rotina utilizará**

...

Restaura da pilha os registradores que a rotina utilizou

RETI

Como recomendações básicas na elaboração dessas rotinas tem-se:

- **Elas devem ser pequenas, com um mínimo de processamento:** rotinas com muitas instruções podem demorar excessivamente, podendo ocorrer outra interrupção antes do seu término. Deve-se, preferencialmente, indicar a ocorrência de algum fenômeno em uma variável (*flag*), para ser considerado posteriormente pelo programa principal.

- **Salvar na pilha apenas os registradores efetivamente utilizados pela subrotina (A, B, PSW), restaurando-os na ordem inversa** (a pilha segue o esquema *Last in, First out* – LIFO**).** Deve-se tomar o cuidado de não se esquecer de retirar o que foi colocado na pilha, na ordem correta (erro comum). O não salvamento dos registradores utilizados tanto pela subrotina de tratamento de interrupções como pelo programa principal pode levar a comportamentos estranhos no programa, nem sempre de fácil descoberta. Lembrar que uma interrupção habilitada pode ocorrer em qualquer ponto do programa principal em execução. Quando este não pode ser interrompido, há a possibilidade de desabilitação das interrupções (colocando-se 0 em **IE.7**).

- **Reservar um dos quatro bancos de registradores para uso pela subrotina,** pois a troca é rápida, através do registrador **PSW (D8H),** representado a seguir.

BIT	PSW.7	PSW.6	PSW.5	PSW.4	PSW.3	PSW.2	PSW.1	PSW.0
	CY	AC	F0	RS1	RS0	OV		P
END.	DFH	DEH	DDH	DCH	DBH	DAH	D9H	D8H
	CARRY	CARRY AUX.	USER DEF.	00: BANCO 0 01: BANCO 1 10: BANCO 2 11: BANCO 3		OVERF.	EXP. FUT.	PARID.

3.3. Os Timers do 80C51

O 80C51 possui dois timer internos programáveis, que podem ser utilizados para diversas finalidades, como por exemplo, contagem de tempo ou eventos, geração da taxa de comunicação do canal serial (necessariamente o Timer 1), gerador de taxas de amostragem e relógio de tempo real. Várias configurações são previstas, bem como o uso ou não de interrupções.

Cada família de microprocessadores apresenta um tipo de implementação para as suas interrupções, sendo encontradas muitas particularidades não necessariamente encontradas em outras famílias. Recomenda-se, nesse ponto, que seja efetuada uma pesquisa sobre as interrupções do 80C51 em alguma(as) das referências: [17], [16] (Application Builder), [13], [14], [15], Capítulo 2 de [1], Nota AN420 de [2], Capítulo 6 de [8], Capítulo 6 de [9], Páginas 10 e 11 de [7] no tocante aos nomes de bits e registradores para configurações em geral. A seguir, serão resumidos os principais conceitos envolvidos.

O tutorial encontrado em [17] (http://www.8052.com/tuttimer.phtml#How%20Count), e outras referências devem ser consultadas para o seu correto entendimento ([1], [12], [13], [14]).

4. BIBLIOGRAFIA

[1] PHILIPS; 80C51-Based 8-Bit Microcontrollers – Data Handbook IC20, Philips Electronics North America Corporation, USA, 1997.

[2] PHILIPS; Application Notes and Development Tools for 80C51 – Data Handbook, Philips Electronics North America Corporation, USA, 1997.

[3] GOMI, Edson Satoshi; Apostila da Experiência Microprocessadores, Disciplina PCS 308 - Laboratório Digital II, 1998.

[4] HIRAKAWA, A.R; CUGNASCA, C.E. Apostila da Experiência Familiarização com a Placa Experimental de Microcontrolador 8051, Disciplina PCS 2497 - Laboratório de Processadores I, 2006.

[5] CUGNASCA, C.E.; HIRAKAWA, A.R. Apostila da Experiência Interface com Teclado e Display, Disciplina PCS 2497 - Laboratório de Processadores I, 2007.

[6] MATSUNAGA, A.M.; TSUGAWA, M.O. Sistema de Pesagem Dinâmica. Projeto de Formatura (disciplina PCS-588). Escola Politécnica da USP, 1997.

[7] 2500AD Software 8044/51; X8051 Cross Assembler User Manual.

[8] SILVA JÚNIOR, V.P. Aplicações Práticas do Microcontrolador 8051 – Hardware & Software, Editora Érica Ltda., 1990.

[9] SILVA JÚNIOR, V.P. Aplicações Práticas do Microcontrolador 8051 – Teoria Geral, Editora Érica Ltda., 1994.

[10] ALFACOM. Módulos Multi-Matrix – Manual de Utilização.

[11] MC8051 – Diagrama Lógico da Placa Experimental do 8051.

[12] INTEL HOME PAGE. Programa Application Builder - ApBUILDER. http://developer.intel.com/design/builder/apbldr/.

[13] MACKENZIE, I.S. The 8051 Microcontroller. Prentice Hall, 3rd edition, 1999. ISBN: 0-13-780008-8.

[14] STEWART, J.W. The 8051 Microcontroller. Prentice Hall, 1993. ISBN: 0-13-584046-5.

[15] INTEL. Embedded Microcontrollers Intel Datasheet. 1995.

[16] INTEL HOME PAGE. ApBUILDER http://developer.intel.com/design/builder/apbldr/.

[17] 8052 TUTORIAL. Página sobre a Família 8051/8052. http://www.8052.com/.

5. PARTE EXPERIMENTAL

5.1 Familiarização com Interrupções e o Timer

Antes de iniciar o planejamento desta experiência, é fundamental a familiarização com os mecanismos de interrupção e a programação do Timer 0 do microprocessador 80C51, através das referências já indicadas.

Pede-se que se incorpore no Planejamento:

- um resumo do funcionamento dos *timers* do 80C51.
- a explicação dos modos de contagem de 8 e 16 bits. O que significa o recurso de *auto-reload*? Em que modo ele se encontra presente? O contador de cada *timer* é do tipo *up* ou *dowm*?
- O *clock* do microcontrolador é **11.0592Mhz**: apresentar a fórmula de cálculo para que seja gerada uma interrupção de um dos *timers* a cada **centésimo de segundo**. Nessa situação, calcular o valor a ser carregado no contador, para o modo 1 e o modo 2.

Para efeito de familiarização, desenvolver os **pequenos** programas de teste:

Programa 1: Timer 0, modo 1 com *Wait for Flag*.

- Programar o Timer 0 no **modo 1** (explicar o funcionamento), para dividir o *clock* de entrada o timer por 100, **sem** o uso de interrupção
- Quando for detectado o final da contagem (geração do sinal de *overflow*), inverter o bit **P1.0**, observando-o com osciloscópio (desconectar o cabo do teclado da Porta 1).

Programa 2: Timer 0, modo 1 com Interrupção.

- Modificar o programa anterior para operar **com** a interrupção. Explicar o significado de **TF0**.

Programa 3: Timer 0, modo 2 com Interrupção.

- Modificar o programa anterior para operar com a interrupção gerada pelo Timer 0 operando no **modo 2** (explicar o funcionamento).

Comentar as diferenças de cada programa de teste.

5.2 Implementação de um Relógio Digital

Implementar um Relógio Digital que apresente no Display os minutos, segundos e centésimos de segundo, na forma "**mm:ss:cc**", fazendo-se uso da interrupção do Timer 0 (**TF0)** para gerar a base de tempo. Siga as seguintes etapas para o projeto, implementação e teste do programa:

a) Utilizar as subrotinas de Display já desenvolvidas na experiência anterior [5] para o mostrador. Teste a subrotina de inicialização e de escrita no Display para a o formato "**mm:ss:cc**".

b) Usar como fonte de referência de tempo os centésimos de segundo, provenientes do Timer 0.

c) Quando do desenvolvimento da subrotina para tratamento de interrupção, deve-se lembrar de que há um endereço correto onde esta rotina deve estar carregada. Descreva a rotina e indique o endereço de carga. Faça o teste da rotina de tratamento de interrupção enviando, por exemplo, um caractere para o terminal (utilizar a subrotina **CO** já utilizada em experiência anterior [4].

d) Na rotina de tratamento de interrupção do Timer 0, introduzir um pequeno código para gerar uma onda quadrada no bit 0 da Porta 1 (**P1.0**). Desconectar o cabo do teclado da Porta 1 e, utilizando o osciloscópio, verificar o correto tratamento da interrupção e o valor da base de tempo.

e) Incorporar os módulos desenvolvidos anteriormente para implementar o Relógio Digital.

5.3 Cronômetro

a) Reconectar o cabo do teclado a Porta 1.

b) Com os conhecimentos e recursos de utilização do teclado já vistos [5] e os resultados do item 5.2, pede-se que seja implementado um cronômetro digital que:

- Disponha de uma tecla para "**zerar**" o cronômetro.
- Disponha de uma tecla para "**disparar**" a contagem de tempo pelo cronômetro, que irá mostrando o tempo decorrido desde seu acionamento.
- Disponha de uma tecla para "**parar**" o cronômetro, mostrando o tempo decorrido entre dois eventos.
- Opcionalmente possua outros recursos, a critério de cada grupo.

5.3 Observações

O planejamento deverá apresentar:

- Descrição do projeto, relacionando suas características principais.
- Especificação de cada uma das subrotinas utilizadas e que serão testadas, **com diagramas estruturados** e linhas de programas com **explicação** sobre qual a função da execução de cada comando. **Planejamento sem esta documentação não será aceito!**